해쉬 알고리듬 표준 HAS-160의 저면적 하드웨어 구현

김해주* · 전흥우** · 신경욱*

A Small-Area Hardware Implementation of Hash Algorithm Standard HAS-160

Hae-ju Kim* · Heung-Woo Jeon** · Kyung-Wook Shin*

이 연구는 2009년도 금오공과대학교 학술연구비에 의하여 연구된 논문입니다.

요 약

임의의 길이의 메시지를 160 비트의 해쉬(hash) 코드로 압축하는 한국형 해쉬 알고리듬 표준 HAS-160의 하드웨어 구현에 대해 기술한다. 저면적 구현과 고속 연산을 위해 단계연산 회로를 5:3 및 3:2 캐리보존 가산기(carry-save adder)와 캐리선택 가산기(carry-select adder)의 혼합구조를 사용하여 설계하였다. 512 비트 메시지 블록으로부터 160 비트의 해쉬코드를 생성하는데 82 클록주기가 소요되며, 50 MHz@3.3-V로 동작하는 경우 312 Mbps의 성능을 나타낸다. 설계된 HAS-160 프로세서는 FPGA 구현을 통해 기능을 검증하였으며, 0.35-㎛ CMOS 셀 라이브러리로 합성한 결과 약 17,600개의 게이트와 약 1 ㎟의 면적으로 구현되었다.

ABSTRACT

This paper describes a hardware design of hash function processor which implements Korean Hash Algorithm Standard HAS-160. The HAS-160 processor compresses a message with arbitrary lengths into a hash code with a fixed length of 160-bit. To achieve high-speed operation with small-area, arithmetic operation for step-operation is implemented by using a hybrid structure of 5:3 and 3:2 carry-save adders and carry-select adder. It computes a 160-bit hash code from a message block of 512 bits in 82 clock cycles, and has 312 Mbps throughput at 50 MHz@3.3-V clock frequency. The designed HAS-160 processor is verified by FPGA implementation, and it has 17,600 gates on a layout area of about 1 ㎟ using a 0.35-㎛ CMOS cell library.

키워드

해쉬 알고리듬, HAS-160, 인증, 정보보호

Key word

Hash algorithm, HAS-160, authentication, information security

* 금오공과대학교 전자공학부

** 금오공과대학교 전자공학부 (교신저자)

접수일자 : 2009. 11. 16

심사완료일자 : 2010. 01. 03

Ⅰ. 서 론

오늘날 정보통신기술과 인터넷의 발전으로 인해 인터넷 뱅킹, 전자상거래, 전자화폐 등의 서비스가 보편화되고 있으며, 이에 따라 유·무선 네트워크를 통해 유통되는 개인정보와 금융정보의 보안을 위해 다양한 정보보호 기술이 적용되고 있다. 그러나 컴퓨터 성능의 향상과 보안공격기술의 발달로 인해 정보 유출과 위조를 막기 위한 더 높은 수준의 암호화 기술이 지속적으로 요구되고 있다. 메시지 내용 공개나 트래픽 분석과 같은 수동적 공격은 DES(Data Encryption Standard), AES (Advanced Encryption Standard) 등의 대칭키 암호 알고리듬과 RSA(Rivest-Shamir-Adleman), ECC(Elliptic Curve Cryptography) 등의 공개키 암호 알고리듬으로 대처할 수 있다. 그러나 단순한 메시지 암호화만으로는 위장, 재전송, 메시지 불법 수정과 같은 능동적 공격에 대해 대처할 수 없으며, 능동적 공격에 대처하기 위한 방법으로 해쉬함수를 이용한 인증(authentication), 무결성 (integrity) 검증, 부인방지(non-repudiation) 등이 사용된다[1].

해쉬함수 알고리듬은 임의의 길이의 비트 열을 고정된 길이의 출력 값인 해쉬코드로 압축시키는 암호 알고리듬의 일종이며, 정보처리 시스템 또는 정보통신망 환경에서 인증, 무결성 검증 및 부인방지 등의 정보보호 서비스를 제공하는 전자서명에 폭넓게 활용된다. 정보보호 응용에 사용되는 해쉬함수 알고리듬은 다음과 같은 세가지 특성을 만족해야 한다. 첫째, 주어진 해쉬코드에 대해 입력 메시지를 찾는 것이 계산상 불가능해야 하며, 둘째, 주어진 입력 메시지에 대해 동일한 해쉬코드를 출력하는 또 다른 입력 메시지를 찾아내는 것이 계산상 불가능해야 하며, 셋째, 강한 충돌 저항성(collision resistance) 즉, 동일한 출력을 갖는 임의의 서로 다른 두 입력 메시지를 찾는 것이 계산상 불가능한 특성을 가져야 한다. 해쉬 알고리듬의 충돌 저항성은 전자서명에서 무결성 검증, 부인방지 등의 서비스를 제공하기 위한 필수적인 요구조건이 된다.[2,6]

해쉬함수는 크게 블록암호 알고리듬에 기초한 해쉬 알고리듬과 전용 해쉬 알고리듬으로 구분할 수 있다. 블록암호를 이용한 해쉬 알고리듬은 DES, AES 등의 블록암호 기술을 사용할 수 있다는 장점이 있으나, 일반적으로 블록암호 알고리듬들은 속도가 느리고 큰 하드웨어 면적을 필요로 하는 단점이 있다. 따라서 대부분의 응용에서는 HAS-160, SHA-1, MD5 등의 전용 해쉬함수 알고리듬이 사용된다.[4-9] HAS-160은 한국형 디지털 서명 표준인 KCDSA(Korea Certification-based Digital Signature Algorithm)에서 사용할 목적으로 개발되었다.[2]

본 논문에서는 한국형 해쉬함수 표준인 HAS- 160 알고리듬의 저면적 하드웨어 설계에 관해 기술한다. 스마트 카드와 같이 동작속도 보다는 작은 칩면적이 중요한 분야에 사용될 수 있도록 저면적 구현에 초점을 두어 설계하였다. 2장에서는 HAS-160 알고리듬을 소개하고, 3장에서 HAS- 160 프로세서의 하드웨어 설계에 대해 기술한다. 4장에서는 설계된 프로세서의 FPGA 구현 검증, 레이아웃 설계 및 성능 분석을 기술하고, 5장에서 결론을 제시한다.

Ⅱ. HAS-160 해쉬 알고리듬

HAS-160 해쉬 알고리듬은 1998년 10월에 국내 표준화(TTAS.KO-12.0011/R1)를 거쳐 2005년 12월에 개정되었다.[2] HAS-160은 임의의 길이의 비트 열을 512 비트의 블록 단위로 처리하여 160 비트의 해쉬코드를 출력한다. HAS-160 알고리듬은 메시지 M을 해쉬함수에 적용하기 위한 메시지 덧붙이기, 메시지 블록 생성, 단계연산 및 최종 해쉬코드 갱신의 과정으로 구성되며 전체 연산 과정은 그림 1과 같다.

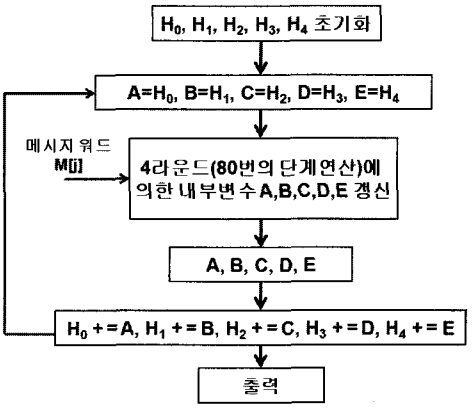


그림 1. HAS-160 해쉬 알고리듬의 연산과정
Fig.1 Arithmetic flow of HAS-160 hash algorithm

HAS-160 해쉬 알고리듬은 5개의 32 비트 워드들 사이의 정수연산으로 구성된다. 입력 메시지는 512 비트 단위의 블록으로 분할된 후, 리틀 앤디안(little endian) 방식의 4 바이트(32 비트) 워드 16개로 변환되어 연산에 사용된다. 입력 메시지 M의 비트 수가 512의 정수배가 되지 않는 경우에는 메시지 덧붙이기를 통해 512 비트의 정수배로 만들어 진다. 입력 메시지를 512 비트 블록으로 분할할 때, 마지막 블록의 길이가 448 비트 보다 작은 경우에는 1 뒤에 0을 필요한 수만큼 채우고 마지막 64 비트에 메시지 덧붙이기 전의 입력 메시지의 길이를 264 법으로 계산한 정수값을 채운다. 마지막 블록이 448 비트 이상일 경우에는 마찬가지로 1 뒤에 0을 필요한 수만큼 채우고 512 비트 블록을 하나 더 생성 하여 마지막 64비트에 덧붙이기 전의 메시지의 길이를 채운다. 그림 2는 마지막 블록이 448 비트 이하인 경우의 입력 메시지를 512 비트의 메시지 블록으로 확장한 예를 보인 것이다.

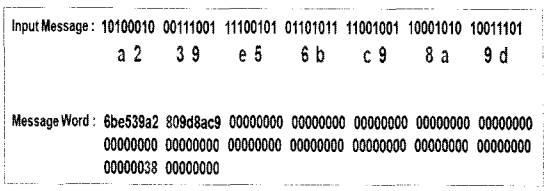


그림 2. 56 비트 입력에 대한 512 비트 메시지 블록 생성
Fig. 2 Generation of 512-bit message word for 56 bits input

HAS-160은 4번의 라운드 연산을 통해 해쉬코드를 생성하며, 라운드 연산은 20번의 단계연산으로 구성된다. 단계연산은 식(1)과 같이 정의되며, 그림 3에서 보는 바와 같이 부울함수 $f[j]$의 계산, 2개의 순환이동(cyclic shift), 4개의 modulo-2^{32} 가산, 내부변수(A, B, C, D, E)의 갱신 등으로 이루어진다.

$$
\begin{aligned}
A_j &= A_{j-1}^{\lll S_1} + f_j(B_{j-1}, C_{j-1}, D_{j-1}) + E_{j-1} + M_j + K_j \\
B_j &= A_{j-1} \\
C_j &= B_{j-1}^{\lll S_2} \\
D_j &= C_{j-1} \\
E_j &= D_{j-1} \text{ (단, } 1 \le j \le 79) \qquad (1)
\end{aligned}
$$

512 비트의 메시지 M으로부터 생성되는 16개의 32 비트 워드 $M[0]$~$M[15]$와 식(2)에 의해 생성되는 4개의 워드를 포함하여 20개의 워드 $M[j]$가 80번의 단계연산에 사용된다.

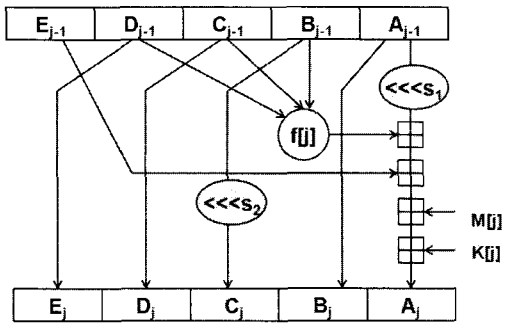


그림 3. HAS-160 알고리듬의 단계연산
Fig. 3 Step-operation of HAS-160 algorithm

$$
\begin{aligned}
M_i[16] = {} & M_i[l(j+1)] \oplus M_i[l(j+2)] \\
& \oplus M_i[l(j+3)] \oplus M_i[l(j+4)] \qquad (2\text{-a}) \\
M_i[17] = {} & M_i[l(j+6)] \oplus M_i[l(j+7)] \\
& \oplus M_i[l(j+8)] \oplus M_i[l(j+9)] \qquad (2\text{-b}) \\
M_i[18] = {} & M_i[l(j+11)] \oplus M_i[l(j+12)] \\
& \oplus M_i[l(j+13)] \oplus M_i[l(j+14)] \qquad (2\text{-c}) \\
M_i[19] = {} & M_i[l(j+16)] \oplus M_i[l(j+17)] \\
& \oplus M_i[l(j+18)] \oplus M_i[l(j+19)] \qquad (2\text{-d})
\end{aligned}
$$

각 라운드 연산에 사용되는 부울함수 $f[j]$와 상수 $K[j]$는 표 1과 같이 정의되고, 순환이동의 크기 (S_1, S_2)는 표 2와 같이 정의된다. S_1은 라운드 마다 동일한 값이 사용되며, 라운드를 구성하는 단계연산에 따라 크기가 달라진다. S_2는 연산단계에 무관하게 동일한 값이 사용되나 라운드 마다 다른 값이 사용된다. 한편, 각 라운드의 연산단계에 적용되는 메시지 워드 $M[j]$의 순서는 표 3과 같이 정의된다. 512 비트의 메시지 블록에 대해 4 라운드에 걸친 80번의 단계연산이 끝나면, 그 결과는 내부 레지스터에 저장되어 다음 메시지 블록의 연산에 사용된다.

표 1. 부울함수 $f[j]$ 및 상수 $K[j]$
Table 1. Boolean function $f[j]$ and constant $K[j]$

라운드 (i)	단계연산 (j)	$f[j]$	$K[j]$
1	$0 \le j \le 19$	$(B \wedge C) \vee (\overline{B} \wedge D)$	00000000
2	$20 \le j \le 39$	$D \oplus C \oplus B$	5A827999
3	$40 \le j \le 59$	$C \oplus (B \vee \overline{D})$	6ED9EBA1
4	$60 \le j \le 79$	$D \oplus C \oplus B$	8F1BBCDC

표 2. 순환이동의 크기 S_1 및 S_2
Table 2. Cyclic shifting values S_1 and S_2

라운드 (i)	단계연산 (j)	S_1	S_2
1	$0 \le j \le 19$	5, 11, 7, 15, 6, 13, 8, 14, 7, 12, 9, 11, 8, 15, 6, 12, 9, 14, 5, 13	10
2	$20 \le j \le 39$		17
3	$40 \le j \le 59$		25
4	$60 \le j \le 79$		30

표 3. 메시지 워드 $M[j]$의 연산 순서
Table 3. Arithmetic order of message word $M[j]$

라운드 (i)	단계연산 (j)	$M[j]$
1	$0 \le j \le 19$	18, 0, 1, 2, 3, 19, 4, 5, 6, 7, 16, 8, 9, 10, 11, 17, 12, 13, 14, 15
2	$20 \le j \le 39$	18, 3, 6, 9, 12, 19, 15, 2, 5, 8, 16, 11, 14, 1, 4, 17, 7, 10, 13, 0
3	$40 \le j \le 59$	18, 12, 5, 14, 7, 19, 0, 9, 2, 11, 16, 4, 13, 6, 15, 17, 8, 1, 10, 3
4	$60 \le j \le 79$	18, 7, 2, 13, 8, 19, 3, 14, 9, 4, 16, 15, 10, 5, 0, 17, 11, 6, 1, 12

Ⅲ. HAS-160 프로세서 설계

3.1 설계사양 및 전체 구조

본 논문의 HAS-160 해쉬 프로세서는 스마트카드에 내장되는 마이크로프로세서의 암호 보조프로세서로 사용될 수 있도록 설계되었다. 32 비트 입출력 인터페이스를 가지며, 입력 메시지의 비트 열과 워드 열 사이의 변환과 메시지 덧붙이기는 마이크로프로세서에서 전처리되어 입력된다. 설계된 HAS-160 프로세서는 전처리된 512 비트의 입력에 대한 160 비트의 해쉬코드를 계산한다.

설계된 HAS-160 해쉬 프로세서의 전체 구조는 그림 4와 같으며, 512 비트의 입력 메시지를 받아 4개의 워드를 추가하여 20개의 워드 열을 생성하는 M_Gen 블록, 80번의 단계연산을 수행하는 HAS160_round 블록, 해쉬코드 초기값을 저장하는 LUT 블록, 그리고 유한상태머신(FSM)의 제어블록 등으로 구성된다.

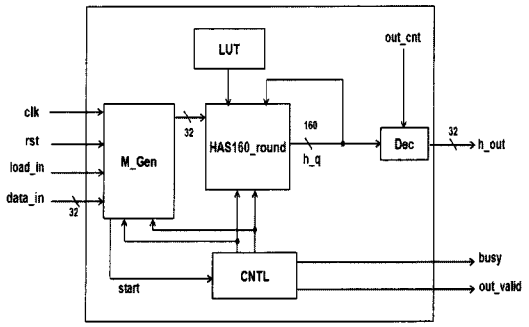


그림 4. HAS-160 프로세서의 구조
Fig. 4 Architecture of HAS-160 processor

3.2 메시지 생성 블록(M_gen)

M_Gen 블록은 그림 5와 같이 20개의 32 비트 메시지 워드를 저장하는 레지스터, MUX, XOR 및 내부 제어회로 등으로 구성된다.

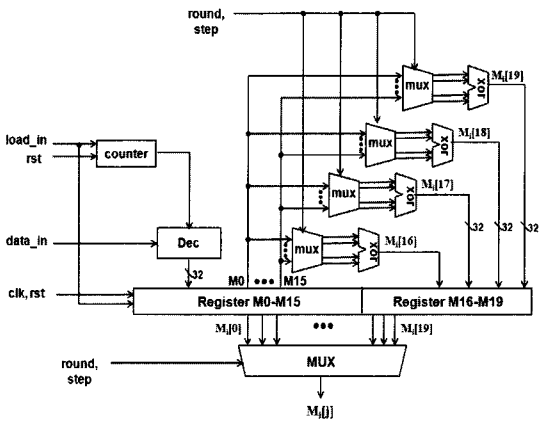


그림 5. M_Gen 블록
Fig. 5 M_Gen block

512 비트의 입력 메시지로부터 16개의 32 비트 메시지 워드 $M[15:0]$를 생성하며, HAS-160 표준문서[2]에 규정된 방식에 의해 16개의 메시지 워드 중 일부를 선택하여 XOR 연산을 통해 4개의 메시지 워드 ($M_i[16:19]$)를 생성하여 레지스터에 저장한다.

3.3 단계연산 블록(HAS160_Round)

단계연산 블록은 표 1에 정의된 부울함수 $f[j]$를 계산하는 f_func 회로, 라운드 연산에 사용되는 상수 $K[j]$을 결정하는 k_init 회로, 표 2에 정의된 순환이동 크기를 결정하는 s1_rotl 및 s2_rotl 회로, 그리고 modulo-2^{32} 가산기 등으로 구성되며, 그림 6과 같이 설계되었다. 제어회로에서 입력되는 라운드 계수기의 값에 따라 $f[j]$와 $K[j]$, 순환이동 크기 S_2가 결정되며, 단계 계수기의 값에 따라 순환이동 크기 S_1이 결정된다. 그림 6의 단계연산 블록에서 연산속도에 가장 큰 영향을 미치는 부분은 다음 단계연산에 사용될 내부변수 A를 갱신하는 가산기 회로이다. 본 논문에서는 5:3 및 3:2 carry-save 가산기와 carry-select 가산기의 혼합구조를 사용하여 설계하였으며, 이를 통해 최소의 게이트로 고속 동작하도록 하였다.

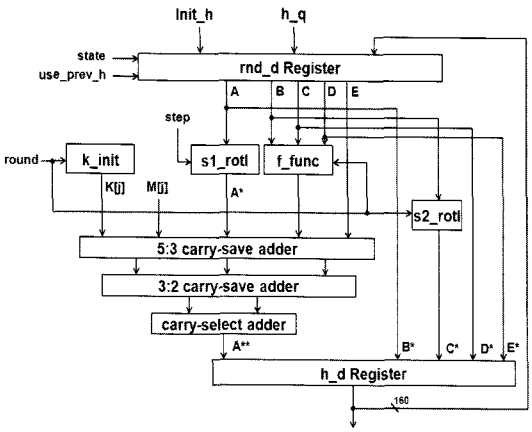


그림 6. 단계연산 블록
Fig. 6 Step-operation block

3.4 제어회로

제어회로는 라운드 및 단계 계수기와 FSM으로 구성된다. FSM은 M_Gen 블록에서 생성되는 start 신호를 감지하기 전의 IDLE 상태, 단계연산을 수행하는 CALC 상태, 160 비트의 해쉬코드를 갱신하고 h_d 레지스터에 저장하는 VALID_OUT 상태로 구성된다. 입력 메시지가 512 비트 이상인 경우에는 use_prev_h 신호를 이용하여 h_d 레지스터를 갱신된 값으로 유지시켜 다음 메시지 블록의 연산에 사용되도록 하였다. 라운드 계수기와 단계 계수기는 FSM의 CALC 상태에서 동작하며, 단계연산 블록의 부울함수, 순환이동 크기, 상수 값 등을 결정한다.

한 클록 당 하나의 단계연산이 수행되도록 설계되었으며, 라운드 당 20번의 단계연산으로 구성되는 4라운드의 연산에 총 80 클록이 소요된다. 512 비트의 입력 메시지로부터 4개의 워드 열을 추가로 생성하는데 한 클록이 소요되며, 단계연산 종료 후 h_d 레지스터의 초기값을 내부변수와 더하여 최종 해쉬코드를 생성하는데 한 클록이 소요된다. 따라서 512 비트의 메시지로부터 160 비트의 해쉬코드를 생성하는데 총 82 클록 사이클이 소요되도록 설계하였다.

Ⅳ. 설계 검증 및 성능 평가

4.1 기능 검증

HAS-160 해쉬 프로세서는 Verilog HDL로 설계되었으며, HAS-160 표준문서에 제시된 1,024 비트의 테스트 벡터를 사용하여 기능을 검증하였다.

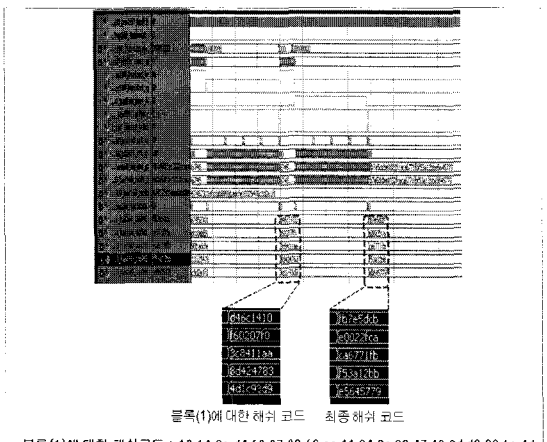
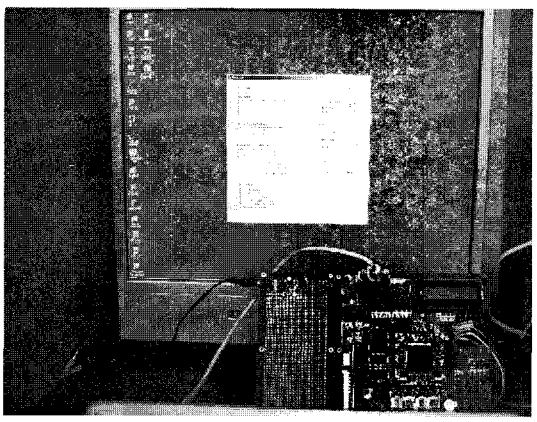


그림 7. 기능검증 결과
Fig. 7 Functional simulation result

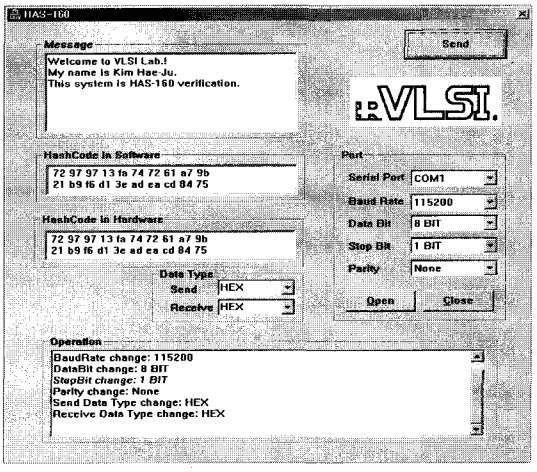
1,024 비트의 테스트 벡터는 512 비트의 블록 2개로 분할되어 순차적으로 연산에 사용되며, 첫 번째 512 비트의 메시지 블록으로부터 160 비트의 해쉬코드가 출력된다. 첫 번째 블록에 대한 해쉬코드는 두 번째 메시지 블록의 연산에 사용되어 최종 160 비트의 해쉬코드가 생성된다. 시뮬레이션을 통한 기능검증 결과는 그림 7과 같으며, 표준문서에 제시된 해쉬코드와 일치하는 결과가 얻어져 설계된 프로세서가 올바로 동작함을 확인하였다.

4.2 FPGA 구현 및 동작 검증

설계된 HAS-160 프로세서는 FPGA 구현을 통해 하드웨어 동작을 검증하였으며, 그림 8은 검증 시스템의 구성과 동작화면을 보이고 있다. Spartan3 FPGA 보드에 HAS-160 프로세서와 Wrapper를 구현하고, UART 포트를 통해 PC와 연결하여 메시지와 해쉬코드가 전송되도록 하였다. MFC를 이용한 GUI를 통해 검증결과가 모니터에 표시되도록 테스트 환경을 구성하였다. 임의의 길이의 메시지를 입력하면, UART 포트를 통해 FPGA 검증보드로 전송되어 해쉬코드가 생성되고, 생성된 해쉬코드는 다시 PC로 전송되어 화면에 표시된다. 한편, PC에서는 HAS-160의 소프트웨어 연산결과를 생성하여 FPGA에서 생성된 결과와 비교한다. 그림 8-(b)는 검증 시스템의 동작화면을 보이고 있으며, FPGA에 구현된 HAS- 160 프로세서의 해쉬코드와 PC의 소프트웨어로 계산된 해쉬코드가 동일함을 확인할 수 있다.



(a)



(b)

그림 8. FPGA 구현 검증 시스템 구성도 및 검증 결과
(a) FPGA 검증 시스템 (b) FPGA 검증 결과
Fig. 8 FPGA verification system and result
(a) FPGA verification system
(b) FPGA verification result

4.3 레이아웃 설계 및 성능 평가

설계된 해쉬 프로세서는 0.35-㎛ CMOS 라이브러리를 사용하여 50 MHz의 동작 주파수로 논리 합성하였으며, 그 결과 총 게이트 수는 17,600이고, 512 비트의 입력으로부터 160 비트의 해쉬코드를 생성하는데 312 Mbps의 성능을 가진다. Astro 툴을 사용하여 P&R을 수행한 결과의 레이아웃 도면은 그림 9와 같으며, 약 1 ㎟의 면적으로 구현되었다.

표 4는 설계된 HAS-160 프로세서의 성능을 비교한 것이다. 본 논문의 해쉬 프로세서는 문헌 [3] 보다 최대 동작 주파수가 느린 것으로 나타났으나 이는 논리합성에 사용된 공정의 차이로 인한 것이며, 동일한 공정을 사용하여 합성하면 비슷한 동작속도가 얻어질 것으로 예상된다. 본 논문의 해쉬 프로세서는 문헌 [3]의 결과에 비해 약 16%의 적은 게이트 수로 구현되었으며, 따라서 저전력 소모가 요구되는 스마트카드의 보조프로세서에 적합하다. 동일한 동작주파수로 환산하면, 본 논문의 해쉬 프로세서는 문헌 [3]과 동일한 성능을 갖는다.

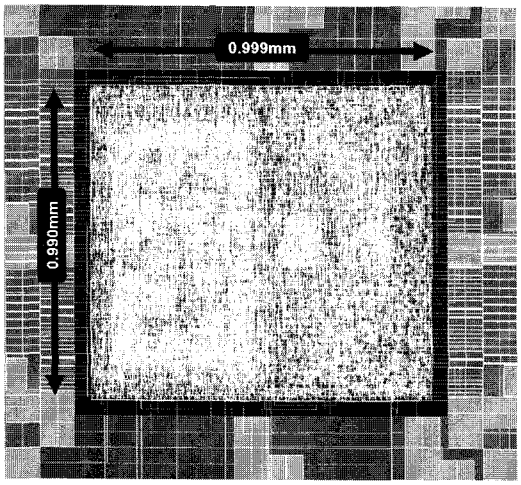


그림 9. HAS-160 프로세서의 레이아웃 도면
Fig. 9 Layout of HAS-160 processor

표 4. 성능 비교
Table 4. Performance comparison

구분	참고문헌 [3]	본 논문
알고리듬	HAS-160	
단계 당 클록 수	1	
총 소요 클록 수	82	
데이터 I/O	32비트	
동작 주파수	175 MHz	50 MHz (Max 75 MHz)
연산 시간 (50 MHz 환산)	467 ns (1.6 us)	1.6 us
게이트 수	21,000 (1.0)	17,600 (0.84)
성능 (50 MHz 환산)	1,093 Mbps (312 Mbps)	312 Mbps
공정	0.25-μm	0.35-μm

V. 결 론

본 논문에서는 한국형 해쉬함수 표준인 HAS-160을 구현하는 프로세서를 설계하였다. 스마트카드 보안 보조프로세서로 사용될 수 있도록 게이트 수의 최소화에 초점을 두어 구현하였으며, FPGA를 이용한 하드웨어 구현을 통해 동작을 검증하였다. 설계된 HAS-160 해쉬 프로세서는 0.35-μm CMOS 공정을 사용하여 50 MHz의 동작주파수로 합성한 결과 17,600개의 게이트로 구현되었으며, 312 Mbps의 성능을 갖는다. 따라서 인증, 무결성 검증 및 부인방지 등의 정보보호 서비스를 제공하는 스마트카드 전자서명 시스템의 하드웨어 구현에 IP로 활용될 수 있을 것이다.

참고문헌

[1] William Staillings, *Cryptography and Network Security,* Principle and Practice, 1999.

[2] TTA, "Hash Function Algorithm Standard (HAS-160)", 12. 2000.

[3] Ju-Dai Hyun, Byeong-Yoon Choi, "Hard- ware Design of HAS-160 Algorithm", Dong-Eui Univ., vol. 37, pp. 415-421, 08. 2002.

[4] 전신우, 김남영, 정용진, "SHA-1과 HAS- 160과 의사난수 발생기를 구현한 해쉬 프로세서 설계", *한국통신학회 논문지,* vol.27, pp. 112-121, 2002.

[5] Yongje Choi, Mooseop Kim, Taesung Kim, Howon Kim, "Low power implementation of SHA-1 algorithm for RFID system", *IEEE Int. Symp. on Consumer Electron ics,* pp. 1-5, June, 2006.

[6] 성수학, "해쉬함수의 최근 동향", *http://mathnet.kaist.ac.kr/real/2006/6/text/sungsuhak.pdf,* June, 2006.

[7] Charanjit S. Jutla and Anindya C. Patthak, "Provably Good Codes for Hash Function Design", *IEEE Trans. on Information Theory,* vol. 55, no. 1, pp. 33-45, Jan. 2009.

[8] SHA-1 Standard, National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-1, *www.itl.nist.gov/fipspubs/fip180-1.htm,* 2003.

[9] Y. Eslaim, A. Sheikholeami, P.G. Gulak, S. Masui, K. Mukaida, "An Area Efficient Universal Cryptography Processor for Smart Cards", *IEEE Trans. on VLSI Systems,* vol. 4, pp. 43-56, Jan. 2006.

※ 반도체설계교육센터(IDEC)의 CAD Tool 지원에 감사드립니다.

저자소개

김해주(Hae-Ju Kim)

2009년 금오공과대학교 전자공학부 (공학사)
2009년 3월~현재 금오공과대학교 전자공학과 석사과정

※관심분야 : 암호 알고리듬, 시스템 및 네트워크 보안, 영상처리

신경욱(Kyung-Wook Shin)

1984년 2월 한국항공대학교 전자공학과 (공학사)
1986년 2월 연세대학교 대학원 전자공학과(공학석사)
1990년 8월 연세대학교 대학원 (공학박사)
1990년 9월~1991년 6월 한국전자통신연구소 반도체 연구단(선임연구원)
1991년 7월~현재 금오공과대학교 전자공학부(교수)
1995년 8월~1996년 7월 University of Illinois at Urbana-Champaign(방문교수)
2003년 1월~2004년 1월 University of California at San Diego(방문교수)

※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계

전흥우(Heung-Woo Jeon)

1980년 2월 한국항공대학교 전자공학과(공학사)
1982년 2월 고려대학교 대학원 전자공학과(공학석사)
1988.8 고려대학교 대학원 전자공학과(공학박사)
1989.3~현재 금오공과대학교 전자공학부 교수

※관심분야 : 집적회로설계, 신경망회로